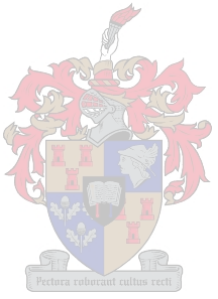
Collection, Evaluation and Selection of Scientific Literature: Machine Learning, Bibliometrics and the World Wide Web

James Connan



Thesis presented in fulfilment
of the requirements for the degree of
Master of Science
at the University of Stellenbosch

Supervisor: Prof. Christian W. Omlin

December 2004

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Abstract

We present a system that uses statistical machine learning to identify and extract bibliography information from scientific literature. Techniques for finding and gathering useful information from the ever growing volume of knowledge on the World Wide Web (WWW), are investigated.

We use hidden Markov models both for recognition of bibliography styles and extraction of bibliographic information with an accuracy of up to 97%. The accuracy with which we are able to extract this information allows us to present a case study in which we apply methods of citation analysis to information extracted from three areas of machine learning. We use this information to identify core sets of papers that have made significant contributions to the fields of hidden Markov models, neural networks and recurrent neural networks.

Opsomming

Ons bied 'n sisteem aan wat gebruik maak van statistiese masjiene wat leer om bibliografiese inligting uit wetenskaplikke literatuur te identifiseer en ontgin. Tegnieke wat aangewend word vir die verkenning en insameling van nuttige inligting vanaf die snel groeiende kennisbron van die WWW, word ondersoek.

Ons gebruik verskuilde Markov modelle vir die herkenning van verwysingsstyl en ontginning van verwysingsinligting met 'n akuraatheidspeil van to 97%. Hierdie hoë ontginningsakuraatheid stel ons in staat om 'n toepassing van die tegniek op die veld van masjiene wat leer toe te pas. Ons rapporteer hoe ons die tegnieke gebruik het om literatuur wat beduidende bydraes in die velde van verskuilde Markov modelle, neurale netwerke en terugkerende neurale netwerke, te identifiseer.

Acknowledgements

I would like to thank:
NEC Research Institute, Princeton (USA) for providing training data;
My adviser, Prof, Christian W. Omlin for his continued patient guidance;
Mr. Reg Dodds, my co-adviser, for the support he provided;
My family, for their support and belief in me;
And my loving wife, who was there for me through all the ups and especially the downs.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

"Frequently cited documents represent the key concepts, methods, or studies in a given field." [4]

1.1 Motivation

Scientific research is seen by many as a cornerstone of progress. Scientific research tends to be concentrated around centres and laboratories that are situated at either institutions of higher education or dedicated research facilities. This concentration of research and researchers allows for the exchange of knowledge through personal contact as well as the sharing of resources. Resources shared include specialized equipment, laboratories, students and libraries.

Libraries provide researchers with access to books, journals and other media which in turn give researchers access to the work done by other researchers. Technological advances and economic changes have had significant impacts on the operation of libraries and the way researchers use libraries.

Increasingly, paper and microfilm media are being complemented or replaced by electronic media. This has advantages such as improved search capabilities and reduced storage requirements. Index cards have also made way for computer based indexing in most libraries.

Reduced costs of printing, reproduction and electronic storage and distribution have also simplified the task of collecting and distributing information.

Researchers can subscribe to journals and newsletters that are rejected by traditional libraries for being too focused.
The motivating factor behind these changes is the Internet, more specific, the World Wide Web (WWW).

1.2 The World Wide Web Changes Scientific Pursuits

In the early 1960s, the United States of America's Defense Advanced Research Projects Agency (DARPA) conceived a computer network to link together their research projects at various institutions. By the end of 1969, four institutions (UCLA, Stanford Research Institute, UC Santa Barbara and the University of Utah) were linked together to form the Advanced Research Projects Agency Network (ARPANET). By using "open architecture networking", ARPANET eventually evolved into what is today known as the Internet. "Open architecture networking" rests on the idea of defining an inter-networking architecture rather than restricting all networks to the same technology.
Devices on the Internet use a protocol called Transmission Control Protocol/Internet Protocol (TCP/IP), developed by Robert E. Kahn and Vint Cerf [1], to communicate with each other. Protocols such as the File Transfer Protocol (FTP) and Hyper Text Transfer Protocol (HTTP) run on top of this underlying TCP/IP network, providing services and functions to end users. The best known of these is the World Wide Web (WWW). The WWW uses HTTP to transfer documents, called web pages, from servers to end user machines. These pages are formatted in a language called Hyper Text Markup Language (HTML). HTML adds meta data to the information contained in the web pages that allow web browsers to render the pages to appear the way the authors of the pages intended it to. The first browser widely used by the public appeared in 1993. It was called Mosaic and ran on the IBM PC, the Macintosh and the X-windowing system.
The number of web pages and Internet users have grown exponentially and it today estimated to be in excess of 1 billion pages and 400 million users [2] [3]. As a result, the volume of information available is beyond comprehension. As hinted at in the previous section, there are vast amounts of academic information available on the Internet and this information is growing daily.

Most researcher, conferences and journals today have a presence on the Internet. Thus both published and unpublished works are increasingly finding their way onto the Internet[28].

1.3 Problem Statement

Researchers venturing onto the WWW will generally use a search engine (discussed in Chapter 2) to attempt to find useful information. They may choose to use a general purpose search engine such as Google (www.google.com) to search the WWW for information about a given topic. Using keywords, results are displayed according ranking strategies determined by the search engine developers. Results typically include information found on researchers' home pages, conference sites, journal sites and may even include results found on special purpose search engine sites. Alternatively a researcher may choose to use a special purpose search engine such as CiteSeer (citeseer.nj.nec.com) or CORA (www.cora.justresearch.com) to search for scientific literature. A keyword search returns links to academic documents ranked using techniques including bibliometrics.

Both general and special purpose search engines return results related to a specific topic rather than a given field. This thesis proposes a method to objectively determine those works that have had a major influence on a given field. It is often desirable for a researcher entering a new field or a new person starting out in research to find a collection of works that give an overview of a given field and its development rather than works dealing with specific applications withing the field. In this thesis we describe techniques that can be used either with existing systems or used to construct a stand alone special purpose search engine.

1.4 Research Hypotheses

Forums that publish scientific research are very strict at regulating the format in which documents must be submitted. This includes dictating how references to cited works must be formatted. If automatic collection and

selection of influential works is to be successful, then the bibliographic information contained in references must be extracted with high accuracy independent of particular styles of referencing. We hypothesize that it is possible to use machine learning techniques, together with accurate bibliographic information to extract the information of cited works from these documents. It is then possible to apply bibliometric methods to determine which works had the greatest influence on a particular field of research.

1.5 Objectives

The objective of this thesis is to show how machine learning can be used to accurately extract information about cited works from formatted scientific documents. We also show how bibliometric knowledge can be applied to this extracted information to objectively determine works that have a major influence on particular fields of study in computer science. The objective is to find works that present information that is relevant to a given field in a useful way, not just works that were ground breaking or introduced new ideas. The fields chosen for this case study include hidden Markov models, neural networks and recurrent neural networks.

1.6 Methodology

We use hidden Markov models (HMMs) to extract information from the references sections of scientific papers. To do this we develop a tool that reads *BibTeX* files and outputs formatted references with additional labelling information included. The HMMs have states corresponding to the fields of the references. They also have additional states that correspond to the transitions between reference fields. This allows the HMMs to not only extract the information contained in the references with a very high degree of accuracy, but to also very accurately discriminate between different referencing styles.

Authors cite papers in which they have found useful information [48] which may include the sources of ideas, support of assertions or methods and designs used in their own work. Thus, by citing a paper, the author suggests to

the reader that the cited paper is relevant to the particular topic under investigation. We determine which papers are cited most often, and therefore had the most influence, in a given field.

1.7 Accomplishments

We designed a system that uses HMMs to extract information form formatted scientific papers. As well as training different models to extract information from different referencing styles, we found it useful to train separate models to extract information form different source types within a given referencing style, i.e. separate models are used to extract information from a reference citing a book or a journal article or a web page, even though all three reference my be formatted using the same style.
We downloaded documents classified as relevant to given topics from *CiteSeer*, extracted citation information from these documents and determined the works that had the greatest influence on the given fields. These results appear at the end of this document.

1.8 Thesis Outline

In Chapter 2, we provide the user with an introduction to the WWW and techniques and tools used to find information on the WWW. We discuss some common techniques used to gather and rank web pages as well specific tool and search engines that use these techniques. We conclude this chapter with a description of such tools that have particular relevance to computer science.
We briefly discuss the field of citation analysis and bibliometrics in Chapter 3. This chapter explores the motivations for citing in academic documents as well as describing ways in which citation information can be used to make useful statements about the documents from which it is extracted.
We introduce citation extraction from scientific literature as a sequence recognition problem in Chapter 4. Since HMMs are ideally suited for this task, we give an introduction to HMMs and describe the evolution of the

models used to extract the relevant information from the scientific documents.

We describe in Chapter 5 the application of the tools and techniques discussed in the previous chapters to three real world examples. The results of our work is summarised and explained in this chapter.

We conclude this thesis with a summary of our results and an overview of suggested future work that can build on what was learned in this thesis.

Chapter 2

Scientific Data Mining on the World Wide Web

2.1 Web Navigation

The Internet is growing at such a rapid rate that it is almost impossible to find reliable information on the exact number of users that use the WWW. Tables 2.1 shows the estimated number of Internet users for the last 5 years [2]. Table 2.2 shows the estimated number of web pages indexed by some of the popular search engines [3].

Date	Number of Users
Sept 2002	605.60 million
May 2002	580.78 million
May 2001	462.62 million
June 2000	336.52 million
May 1999	171.25 million
March 1998	66.68 million

Table 2.1: Estimated growth of the community of Internet users 1998-2002

Uniform Resource Locators (URLs) are used to identify sites on the WWW. An example of a URL is "citeseer.nj.nec.com". A user enters the URL of a desired page into the address bar of the web browser on the local machine. A request is sent from the local machine via the Internet to the machine hosting the desired web page. A copy of the page is then sent from the

Search Engine	Estimate (millions)
Google	3,033
AlltheWeb	2,106
AltaVista	1,689
WiseNut	1,453
Hotbot	1,147
MSN Search	1,018
Teoma	1,015
NLResearch	733
Gigablast	275

Table 2.2: Size of page indices of popular search engines

remote machine to the local machine. The web browser on the local machine then renders the page enabling the user to interact with it.
HTML allows the web site designer to link pages by embedding URLs within pages. If a user clicks on this link, the browser will open the page pointed to by the link. This page can reside on the same site as the document containing the link or it can be on any other site on the WWW. Any page may contain any number of links; by following these links, the user can navigate the WWW. The WWW can be visualised as a graph where nodes and edges represent pages and links between pages, respectively.

2.2 Scientific Information Dissemination on the WWW

Libraries collect information and have traditionally been the main source of information for researchers. There are some disadvantages associated with the use of libraries. Not being in close proximity of a library can severely stifle research. Library hours determine when a researcher can look for more material. Journals or books may be on loan. Even familiarity with the procedures and various indices available may require a still substantial effort to locate the desired material. Material that is not available in the local library can usually be obtained from another library through inter-library loans. This service does, however, come at a price, both financially and time wise. Most of these disadvantages do not apply to the Internet. However, the unstructured nature of the Internet is a major disadvantage.

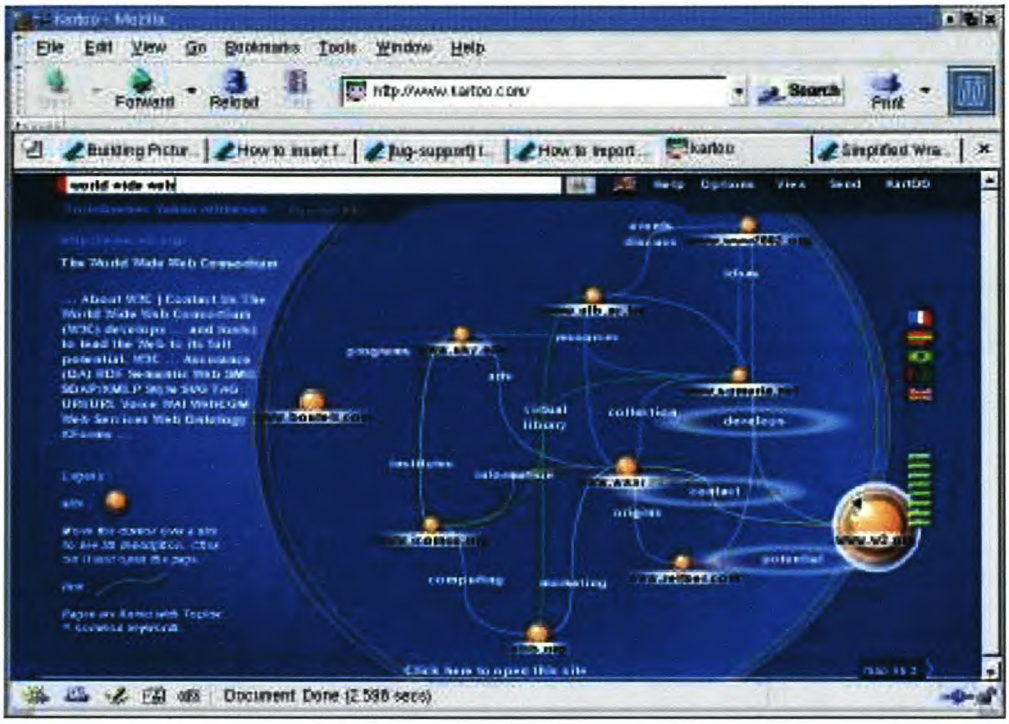


Figure 2.1: Results from a search engine with graphical output showing the links between sites. Note that the links here are key words and not URLs.

The Internet has grown at a phenomenal rate. In April 1998, there was an estimated minimum 320 million pages on the World Wide Web [27]. By February 1999, this figure had grown to 800 million pages [28]. As the number of pages has increased, so has the amount of scientific data available on-line. By 1996, about 1000 journals were available on-line [51]; by 2001, the number of research articles on the WWW exceeded 1 million [26]. There are many ways in which articles find their way onto the WWW. Some conferences and journals provide free on-line access to their articles. In some cases, this access may be time-delayed. Others allow authors to freely either publish their articles on the web or to purchase the right to publish their articles on the web. Some require some sort of renumeration, in the form of membership fee. In [26], Lawrence shows that offline and on-line conference articles in computer science published from 1990 to 2000 received a mean number of citations of 2.74 and 7.03, respectively. Older articles are less likely to be available on-line and may also accumulate more citations due to their longer

availability. It is thus more appropriate to compare the mean number of citations on a per year basis and then average it over all the years. The figure then rises from on-line articles being cite 2.6 more often to 4.5 times more often.

Lawrence et al. also investigated publications that existed both offline and on-line. They defined multiple years for the same conference as separate publishing venues. They found that there were 336% more citations to online articles compared to articles published offline in the same venue. The authors speculate as to possible reasons for these results. For the purpose of this work, we do not wish to comment on factors such as the visibility or quality of works published on-line; rather, we observe that more and more scientific literature is being published on-line.

2.3 Scientific Data Mining on the WWW

Data mining has a traditional association with databases. Databases usually have a neat and structured design that accommodates data mining. The unstructured nature of the WWW places doubts about the success of data mining on the WWW [14]. Most web sites and pages are designed to be human readable and not machine readable. In an attempt to make the WWW more accessible for data mining, suggestions ranging from changing the WWW into a massive layered database [55] to hand coding custom software ("wrappers") that encapsulate pages and allows information extraction [24] have been made. The explosive dynamic growth of the WWW, however, dooms these approaches to failure.

Etzioni [14] argues that the WWW is sufficiently structured for data mining to be successful. Some of the WWW structures that he refers to include linguistic and typographic conventions, HTML annotations (e.g. <title>), classes of semi-structured documents (e.g., product catalogs), web indices and directories. Successful web mining requires resource discovery, information extraction and identification of patterns within collected data, e.g. using machine learning methods.

2.3.1 Resource Discovery

The WWW is an information resource that is growing very rapidly. It is also very dynamic [27]. For these reasons, even established search engines need to continually explore the WWW looking for new content and verifying or updating existing content. Programs called web robots or web crawlers are used to perform this task. We will take a more detailed look at crawlers in Sections 2.4 and 2.5. Search engine developers use crawlers to collect data from the WWW so that they can construct indices that can be used to retrieve information from the WWW. These indices establish a link between a crawled page and keywords contained within the page. By presenting the user with an interface that allows them to perform keyword searches on such an index, it is possible to present the user with a list of web sites that contain the keywords used in the query.

Unfortunately, these indices each only cover a small fraction of all the pages on the WWW. The lists extracted from these indices often contain irrelevant, outdated, or unavailable references. One solution that tries to overcome these problems is MetaCrawler. When a user submits a keyword to MetaCrawler, it in turn submits it to the indices of several different search engines. This feedback is then collated and presented to the user. This attempts to extend the coverage of the search and attempts to reduce irrelevant, outdated, and unavailable links.

Lawrence et al. estimated a lower bound for the number of pages on the WWW by analysing 575 searches performed on 6 different search engines[29]. Their method for estimating the number of pages is as follows:

The fraction p_a of the indexable web covered by search engine a is $p_a = \frac{n_0}{n_b}$, where n_0 is the the number of documents returned by both search engine a and search engine b and n_b is the number of pages returned by search engine b. The estimated size of the indexable web is then $\frac{s_a}{p_a}$ where s_a is the number of pages indexed by search engine a.

They were able to estimate a lower bound for the number of pages on the WWW from the results of the two largest search engines. They could only estimate a lower bound as this technique assumes that the indexing done by the two search engines is completely independent; this is not the case. They found that the search engine that indexed the most pages only indexed about

two thirds of the WWW. When the results from all the search engines were combined, the amount of information presented to the user was 3.5 times more than when using only a single search engine.
Dead links, i.e. links to sites that no longer exist, accounted for 1.6% to 5.3% of returned results, depending on the search engine used. The authors concluded that even if a way could be found to index all pages that do not have links pointing to them, one would have to index all pages on the WWW simultaneously if you wished to construct a truly comprehensive index of the WWW. This is clearly not possible with current technology.

2.3.2 Information Extraction

Information extraction can be done by hand coding programs or scripts that can extract information from structured WWW documents or by using software that automatically learn descriptions of web services. Harvest and FAQ-Finder are examples of such programs. Harvest neither discovers new documents nor does it learn new models; instead, it relies on semi-structured documents to extract information. A demonstration was performed in which it created a directory of toll-free numbers found in documents on the WWW. This was possible because toll-free numbers have a specific structure.
FAQ-Finder [20] and Auto-FAQ [52] searches lists of Frequently Asked Questions (FAQ) found on the WWW to answer queries put to them. Both systems rely on the structured nature of information found in FAQ lists to extract information.
By contrast, the Internet Learning Agent (ILA) [42] and Shopbot [12] are two examples of WWW miners that automatically learn descriptions of web services. Automatic information extraction agents such as Softbot can then be used to extract information from the learned descriptions. ILA performs queries about known objects on unfamiliar resources. By comparing the answers to known queries, it can learn the format of the output and hence how to extract information from it. For example, by submitting a query about a known person to a phone directory service, it can infer the format of the output by comparing it to the known information about the person. Shopbot is similar to ILA. Shopbot is given the URL of a shop's home page and the category of the product desired. It then explores the site looking

for a searchable catalog. It learns the format in which information appears in the catalog and then learns to extract information about products from the page. It uses information about popular products to analyse the information found in the store's catalog. In a preliminary study, it managed to shop faster and get better prices than users with manual WWW navigation [12].

2.3.3 Generalisation

Machine learning systems deployed on the WWW are usually designed to learn about users' preferences rather than learning about the WWW. One of the major obstacle with machine learning is the dependence on labeled data. Machine learning typically requires data labeled as relevant and irrelevant in order to train. The WWW has an abundance of data, but unfortunately the vast majority is not labeled.

Attempts have been made to reduce the amount of labeled data needed for training. [41] uses a combination of the Expectation-Maximisation (EM) algorithm and a naive Bayes classifier. First, a classifier is trained on labeled documents; the trained classifier then probabilistically labels the unlabeled documents. A new classifier is then iteratively trained on all the documents until convergence is reached. The authors were able to reduce classification errors by up to 30% on real world data.

Projects like Ahoy! attempts to use the fact that the WWW is an interactive environment [13]. It queries WebCrawler and then applies heuristics to the output to filter the results. It then asks the user to supply feedback about the processed results. Ahoy! rapidly collects data it needs to learn because it relies on multiple users for feedback.

2.4 Web Crawlers

The Internet contains a vast amount of information; real-time searching by entering URLs into a web browser is very inefficient. Even automating the process in software is not very efficient as each query will generate traffic across the Internet. In order to improve search efficiency, search engines have been developed to help users locate the pages they are looking for.

Some search engines are maintained by academic institutions and are used mainly for research purposes while others are commercial enterprises. One of the fundamental cornerstones of search engines are web indices. An index stores information linking URLs and keywords. When a user submits a query to a search engine, these indices are used to generate results rather than attempting to search the WWW in real-time. The process of traversing the WWW gathering web pages is called crawling. Other terms used to describe this process are trawling, spidering and robot scheduling. The software that does the crawling is called a crawler.

Performing a breadth-first or depth-first crawl of the entire WWW is a very expensive process both in terms of bandwidth and processing power. Considering the volume of pages on the Internet, storing information about each page also requires a lot of storage space. This places severe limitations on the number of developers that can afford to get involved in search engine construction and maintenance. The continual growth of the Internet is therefore a limiting factor in new search engine construction.

In order to protect their commercial interests, most general purpose search engines are reluctant to disclose information about the crawlers that they use. Google [7], one of the more popular search engines, breaks the trend here and is quite well documented. Here is a brief discussion of some of the difficulties they face. The crawler interacts with hundreds of thousands of web and name servers that are maintained by entities that have no connection to Google. Google typically runs a single URL server that supplies URLs to several crawlers. Every crawler has several hundred simultaneous connections to the Internet. A Dynamic Name Server (DNS) is used to translate host names from the URL to Internet Protocol (IP) addresses needed by the network protocol to find the machine that hosts the web site. This causes a significant bottleneck for Google and as a result each of their crawlers maintain its own DNS cache in order to and avoid repeated DNS lookups. The number of pages and servers involved makes debugging very difficult. Google's crawler is designed such that interaction does not cause any negative effects to the servers or itself. Assuring that the crawler does not crash or behave incorrectly is a monumental challenge, especially if one considers the fact that there are so many pages on the WWW.

The dynamic nature of the WWW implies that crawlers need to revisit pages

periodically in order to ensure that the indices are up to date. Analytical approaches which optimise the schedule according to which web crawlers revisit pages have been developed. In [9], Coffman et al. describe one such approach. They show how to minimise the cost function $\Sigma c_i r_i$, where c_i are the weights proportional to the page-change rates and r_i the fractions of time spent out of date and therefore schedule the robot in an optimal fashion.

2.5 Focused Web Crawlers

The size and rapid growth of the WWW means that exhaustively crawling it is beyond the means of researchers and institutions. Focused crawlers were introduced in an attempt to reduce the resources needed to index the WWW. The aim of focused crawlers is to search only that subset of the WWW that is relevant to a particular category. This means that the crawler must evaluate all possible paths from the current position and only crawl those that are relevant to its category. This choice is complicated by the fact that short-term gains may be obvious, while long-term gains may be obscured behind short-term losses. The goal is to design a crawler that will find the maximal set of relevant pages while at the same time traverse the minimal number of irrelevant pages on the WWW. Such crawlers can be used to build niche search engines that require so little bandwidth, storage and processing power that groups and individuals can, and does, use PC based implementations. It is conceivable that niche search engines, based on focused crawlers, will become the method of choice for users searching for information on the WWW. Users then select the search engine to use based on the topics they wish to investigate. CiteSeer (http://citeseer.nj.nec.com/) and Cora (www.cora.justresearch.com) are examples of such niche search engines.

Traditional graph theory forms the foundation of many crawlers, especially exhaustive crawlers. Exhaustive crawlers typically perform a breadth-first search on the WWW, ignoring the content of pages along the way. All pages in the graph are visited. A focused crawler attempts to identify the most promising links, ignoring links that are off-topic. It therefore needs to analyse the content and link structure of pages along the way.

A focused crawler investigates all the documents that the current document

links to. A classifier is used to assign a score to each of the children. Classifiers can be hard coded models or can be intelligent agents that learn to refine themselves using machine learning techniques. The links to the children are then sorted according to the scores assigned to them and placed in a queue. The children are then searched in a best first order by popping the next link to search from the head of the queue.
Another problem faced by focused crawlers is that links on the WWW are usually uni-directional. Therefore, if there is no link from a child to its parent, the parent will not be crawled and valuable information may never be found. An example is that of a researcher whose home page is found via a link from a list of papers at a conference site. If the researcher does not have a link to the department's member site on his home page, then it is very likely that other researcher in the same department that are active in the same field will not be found. This example is from [10] where the authors present a focused crawler that attempts to overcome this and other problems faced by traditional focused crawlers.

2.5.1 Reinforcement Learning

In [45], Rennie et al. discuss the design of a focused crawler as part of the Cora project. The focused crawler uses reinforcement learning [22] to decide which paths to choose. The advantage of using reinforcement learning is that it enables the crawler to choose paths that have long term benefits with no obvious short term benefits. The aim of reinforcement learning is to learn a mapping from states to actions $\pi : S \rightarrow A$, called a policy, where we have a set of states S such that $s \in S$ and a set of actions A such that $a \in A$. We also have a state transition function $T : S \times A \rightarrow S$ and a reward function $R : S \times A \rightarrow \Re$. The value function for each state s while following policy π is then:

$$V^{\pi}(s) = \sum_{t-0}^{\infty} \gamma^{t} r_{t},$$

where $0 \leq \gamma < 1$ devalues future rewards and r_t is the reward received after t steps. The optimal policy $\pi^{\star}$ is the policy that maximises the value of $V^{\pi}(s)$ for all states s. In order to find $\pi^{\star}$ we need to learn its value function $V^{\star}$ and its specific correlate Q. For each state s and action a they define $Q^{\star}(s, a)$

as:

$$Q^\star(s,a) = R(s,a) + \gamma V^\star(T(s,a)).$$

It is possible to define the optimal policy $\pi^\star$ in terms of Q such that $\pi^\star(s) = \arg\max_a Q^\star(s,a)$. The optimal policy $\pi^\star$ is then found using dynamic programming.
The focused crawler trained using this technique was three times as efficient as crawlers using traditional breadth-first searching and twice as efficient as crawlers using reinforcement learning with only immediate reward.

2.5.2 ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighbourhoods for Information Discovery

ARACHNID [38] is a distributed system that uses a genetic algorithm to guide a focused crawler. A list of keywords and starting documents are used to initialise the algorithm. Each agent is initialised with pre-fetched documents and a random behaviour and "energy". The agent analyses the text of the current document and determines a link relevance estimate to neighbouring documents. It then follows a link based on this link relevance estimate. The agent receives positive energy based on the relevance of the document and negative energy for the cost of using the network. No positive energy is accumulated if the document in question has already been visited. The energy acts as a reinforcement signal. Agents can also reproduce or be eliminated.

Menczer [38] shows how this can be implemented using either a naive or a vector representation. Comparison of a focused crawler using this algorithm with the naive representation shows an improvement of between one and two orders of magnitude over a normal breadth-first search.

2.6 General Purpose Search Engines

General purpose search engines are used by users and automated systems to locate pages on the WWW that contain information relevant to certain queries. Search engines are usually accessible through HTML web pages.

Users submit keywords, topics, phrases, questions, etc. to a search engine. The modus operandi for input submission depends on the search engine. Most search engines allow users to search for keywords. Other frequent options include, but are not limited to, allowing exact word of phrase matching, including specific words, or excluding certain words. Some search engines are very sensitive to the input provided and adding or leaving out a single word can have dramatic effects on the search results. Once the user has submitted the search criteria, the search engine presents a query to its index and returns the pages which it deems relevant. These are then displayed on another HTML page. On this page, users may expect to find a brief description of the pages found in the index and a link to the page. The results are usually ranked in order of decreasing relevance. The user can then follow a link to desired pages. Kartoo (www.kartoo.com), for example, returns the results in a graphical form that attempts to both illustrate the rank of the pages as well as the keywords that link them.

One of the first search engines on the WWW was the World Wide Web Worm (WWWW) [35]. In 1994, it indexed about 110 000 web pages and web accessible documents and received about 1500 queries per day. By the end of 1997, search engines claimed to index between 2 million and 100 million pages and receive about 20 million queries per day. This once more illustrates the rapid growth, not only in scope and popularity of search engines, but also the rapid growth of the users base and the Internet itself.

One of the most important functions of a search engine is the ranking of the search results. There may be millions of pages that have some relevance to the search criteria. Users only want those documents that have high relevance to their queries. One of the earliest attempts to address this problem was to allow the user to perform a new search using only the results from the previous search as the "dataset". This approach has largely disappeared in current commercial search engines. Commercial search engines attempt to rank the search results in such a way that pages that contain the requested information is ranked highest. Unfortunately, commercial search engines are very secretive as to how they achieve this. Their concern is that they may compromise their competitive edge by disclosing this information. Ironically, Google, one of the best search engines on the WWW, is well documented. Google uses a system called PageRank to rank their pages. Next follows a

brief overview of some ranking strategies designed for general purpose search engines, starting with PageRank.

2.6.1 PageRank

The PageRank for of a page A is given as follows:

$$PR(A) = (1-d) + d(PR(T_1)/C(T_1) + \ldots + PR(T_n)/C(T_n))$$

where,

A is the page being ranked, $T_1 \ldots T_n$ are the pages that point to A, $C(T_n)$ are the number of links out of T_n, and d is a damping factor between 0 and 1, default 0.85.

PageRank makes the assumption that, if many pages point to a page or if a page or pages with high PageRanks point to it, then the people that build and maintain sites on the WWW thinks it is a good page and therefore it deserves a high ranking.

2.6.2 Boolean Retrieval Model

The Boolean Retrieval Model is based solely on the presence or absence of keywords in a given document. This measure calculates the relevance $R_{i,q}$ of a document i to a query q.

$$R_{i,q} = \sum_{j=1}^{M} C_{ij}$$

where,

$R_{i,q}$ is the relevance score for page i to query q, M is the number of words in the query, C_{ij} is the occurrence of the j-th query word in P_i, where, $C_{ij} = 1$ if P_i contains the j-th query word else $C_{ij} = 0$, P_i is the i-th WWW page.

It is necessary to remove all disjunctions and negations from the query before this method can be used. Normalising or splitting the query into separate conjunctive clauses will get rid of disjunctions. Omitting all pages that contain negated words prior to ranking will get rid of negations.

2.6.3 Boolean Spread Activation

The Boolean Spread Activation extends the Boolean Retrieval Model to consider neighbouring documents. If a document does not contain a particular query word, but is linked to a document that does contain the word, it is treated as if it contains the word, but with a lower weighting. The assumption is that if documents are linked together there must be a semantic relationship between them. The calculation of the relevance $R_{i,q}$ of document i to query q now becomes:

$$R_{i,q} = \sum_{j=1}^{M} I_{ij}$$

where,

$R_{i,q}$ is the relevance score for page i to query q,
M is the number of words in the query,
P_i is the i-th WWW page,
C_{ij} is the occurrence of the j-th query word in P_i and

$$C_{ij} = \begin{cases} 1 & \text{if } P_i \text{ contains the } j\text{-th query word} \\ 0 & \text{otherwise} \end{cases}$$

$$I_{i,j} = \begin{cases} c_1 & \text{if } C_{i,j} = 1 \\ c_2 & \text{if there exists a } k \text{ such that } C_{k,j} = 1 \text{ and } Li_{i,k} + Lo_{i,k} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$Li_{ik} = 1$ if an incoming hyperlink exists from P_k to P_i, else 0
$Lo_{ik} = 1$ if an outgoing hyperlink exists from P_i to P_k, else 0

c_1 and c_2 are positive constants[54].

2.7 Special Purpose Search Engines

Special purpose search engines aim to overcome some of the problems inherent to general purpose search engines. Amongst these problems are the

quality of results returned by general purpose search engines and the cost of indexing the WWW.

Most general purpose search engines can only do keyword searches. A user submits a query to the general purpose search engine and is presented with a list of URLs that match words or phrases in the query. Special purpose search engines try to go further. By extracting meta-data and using domain knowledge, search engines attempt to return knowledge and not just word matches.

Another problem is the cost of building an index. Special purpose search engines use focused crawlers to build indices. Only a small subset of the WWW needs to be indexed if one is only interested in covering a given topic.

2.8 Metasearch Engines

Metasearch engines [39] are search engines that query other search engines. Metasearch engines do not build indices. Instead, they use other search engines to try and resolve queries. When a user submits a query to a metasearch engine, the metasearch engine presents the query to other search engines. It then processes the results from all the queried search engines and presents them to the user. This has several advantages: The search engine does not have to go through the expensive process of building an index. Also, it increases its coverage of the WWW by using several search engines. Combining results from different search engines that use different ranking schemes is not trivial; but before that question can be addressed the decision as to which general purpose search engines to submit the query to needs to be made.

2.9 Related Work

Most WWW users are familiar with at least one general purpose search engine. The same can not be said for niche search engines. In this section, we will introduce the reader to some of the niche search engines that are available on the WWW and have relevance to computer science.

Researchers often use tools and ideas developed by other researchers; thus they need access to each others' work. Access to others' work also helps

to avoid duplication. Unfortunately, it is not always easy to get access to the work done by others. One of the rewards that researchers cherish is the knowledge that they have made a contribution to the body of shared human knowledge. The time lag between the completion of research and the publication in scholarly journals and conference proceedings can have a negative effect on research. A faster distribution medium, such as the WWW, may alleviate some of these problems. Publications on the WWW are often available before their traditional paper counterparts. Some authors also make their work available on the WWW as soon as the research is completed. In this way, the WWW does allow much quicker access to information, especially in rapidly evolving fields. On-line publication is not without pitfalls, though. The ease of publication results in much more information being made available, which in turn means that the researcher has to sift through a lot more information in order to extract the desired high-quality information. Also, papers published on the WWW have not necessarily undergone a rigorous review process. Even though each researcher or institution may in their opinion organise their work logically and sensibly on-line, this organisation varies from site to site making it very difficult for the researcher to navigate through all the information.

The nature of published scientific works is one of the major reasons that general purpose search engine fail to index scientific works. Some scientific works are available on the WWW in HTML format, but must are published in formats such as Postscript. General purpose search engines are optimised to index HTML documents and therefore one cannot expect then to perform as well on documents in other formats, such as Postscript.

2.9.1 CiteSeer

CiteSeer [6] is a system designed to fill this gap left by traditional general purpose search engines. It uses autonomous web agents to gather and identify scientific publications. Users can search the scientific literature that it gathers. Bibliometric information about the literature is used to return useful information to the user.

CiteSeer has three primary aims:

- To automatically find and retrieve publications on the WWW.
- To make the papers searchable.
- To help the user obtain related papers by using similarity, measures derived from semantic features of the retrieved documents.

CiteSeer uses search engines and heuristics to find and download papers. Semantic information, e.g. citations and word frequencies, are extracted from these papers. Users can then perform keyword searches on a database containing the semantic information or use the links between citing and cited documents to retrieve related research.
CiteSeer maintains an up-to-date snapshot of relevant literature found on the WWW. From this, it autonomously constructs citation indices [19]. There are other commercial citation indices available, but those are usually constructed manually. CiteSeer can be seen as a step toward a "universal, [Internet-based] bibliographic and citation database linking every scholarly work ever written", as suggested by Cameron [8]. It can only partly fulfill this task as it only looks at works that are available on the WWW. It requires no effort from the author beyond placing their work on the WWW. CiteSeer maintains a database that contains the following information:

- *document:* Contains pieces of text from the document, the URL of the document, and an Unique Article ID number (UAID)
- *documentwords:* Contains word frequency information about the body of documents referenced in the *document* table.
- *citation:* Contains the text of citations made by the documents in the *document* table as well as parsed field information. Each record in this table has a Unique Citation ID Number (UCID) and a field for the corresponding UAID.
- *citationwords:* Contains word frequency information about the citations in *citation.*
- *citecluster and clusterweights:* Contains cluster number and weight information when grouping identical citations in different forms. This information is used for automatic similar document retrieval.

Heuristics extract the following information:

- *Header:* This is the information at the beginning of the paper that contains the title, author, institution and other information that comes before actual document text. Identification of features inside the header (e.g. author, title) is not performed yet.
- *Abstract:* If it exists, the abstract is extracted.
- *Introduction:* If it exists the first 300 words of the introduction section are extracted.
- *Citation:* The list of references made by the document are extracted and parsed further as described below.
- *Word Frequency:* Word frequencies are recorded for all words in the document except those in the citation and stop words. The recorded words are stemmed using Porter's algorithm.

Heuristics are used to extract the title, author, year of publication, page numbers, and citation tag from the references identified above. After removing stop words and stemming, the word frequency for each citation is recorded. The heuristics used to extract the information from the reference first identifies invariant information, such as a date, and then guesses the remaining subfields. Users interact with the system through an HTML page. This allows the user to exploit the information stored in the database. CiteSeer does not use word counts to measure relationships between documents. Weaknesses of word counts are that they give high scores to uncommon words shared by documents, even if the sharing is coincidental and they give false scores when words are ambiguous.

2.9.2 Cora

Cora [37] is an Internet portal for computer science research papers. It uses a web crawler to gather papers from the WWW. The crawler uses reinforcement learning as described in Section 2.5.1. Machine learning techniques are used to automate the creation and maintenance of Cora.
Once papers have been downloaded and converted into plain text, information is extracted from the header and bibliography sections. This information includes fields such as authors, title, publication journal and date. The extracted information is used to match papers and references. Citations to the

same paper are grouped together. A matching algorithm is used to ensure that duplicate papers are not added to the repository. A text classification algorithm determines where in the computer science hierarchy the paper belongs.

From this extracted information, a search engine is created. The title, authors, institution, references and abstract are used to identify each paper. Stop words and stemming are not used. The query results are ranked using the weighted log of the term frequency, summed over all terms. The weight is the inverse of the word frequency in the entire corpus. Phrases are treated as single terms. Search queries are not expanded.

Different HMM algorithms are used to extract information from both the headers and the references. A HMM with multiple states per class is used. State are reduced by merging and the performance measured after each merge. Both labeled and *distantly-labeled* data are used for training. The distantly-labeled data is obtained from BibTeX files.

Using these techniques, they were able to extract information from headers and references with an error rate of 7.3% and 6.6%, respectively.

Chapter 3

Citation Analysis and Bibliometrics

3.1 Introduction

Citation analysis is frequently used to analyse the nature and practice of both inter-discipline communication and communication within disciplines. It may serve to clarify the intellectual and social connections between cited and citing works. However, it often meets with criticism, especially when used to determine impact and merit in the reward system of higher education. 'Citation analysis has conquered the world of science policy analysis', according to Leydesdorff and Amsterdamska [34].

In this chapter, we briefly explore some of the techniques used in citation analysis. A small subset of these techniques are then identified to be used to rank academic works. *Bibliometrics* is the statistical analysis of books, journals and other published works [43]. Bibliographies can be constructed from references found in published works. These references contain information such as authors, titles, publishers and dates about other works that the author(s) found were in some way relevant to the current work. By applying statistical methods to the bibliographies bibliographers can identify core literatures in academic areas [11]. *Scientometrics* is the application of quantitative methods to the history and sociology of science [18].

Citation analysis is a sub-area of both bibliometrics and scientometrics. Early citation studies were modest in scope [50]. These studies were based

on lists of references located in a very few books and journals. Transcription and manipulation of citations by hand was tedious.
In 1955, Garfield suggested a set of purposes that authors may have for citing particular works [16]. Garfield suggested that the purposes are all symbolic; they may symbolise connections between the content of the citing and cited text or they may symbolise connections that are related to the authors rather than the content. Citation of one article by another indicates related intellectual context and the citation represents past knowledge. This hierarchical organisation of references allows researchers to locate source publications with shared focus that the researcher might be interested in.

3.2 Purpose of Citations

Citing behavior can cause problems. Various biases may enforce the citing of some items or authors and lead to the avoidance of others. Authors may cite their own work with so-called self citations. When analysing citations, the context in which a citation is made, is ignored. An author may cite another author or item because the cited work has relevance to his or her own work. By contrast, the author my be expressing criticism of the cited work. In such a case, the cited work may not be central to the publication. Shadish *et al.* surveyed a set of authors to try and determine their reasons for citing the works which they did [48]. Their findings were as follows:

- 18.1% cited to support an assertion
- 16.1% cited to demonstrate sources of methods or designs that were used in their work
- 9.1% cited classics in the field
- 7.5% claimed that a citation represented a particular genre of study or a particular concept
- 15.6% said the citation was personally motivated. Personal motivations included:
  - it influenced the authors thinking
  - it was the source of an idea

- it justified a central argument
- it was similar to the authors work
- it reviewed the literature

No respondent to the Shadish *et al.* survey admitted to social reasons for citing a particular work. Social reasons include: prestige of cited journal, the possibility that the cited author may review the citing manuscript, or simply to satisfy a potential reviewer's desire to see a demonstrated familiarity with the literature.

3.3 Citation Measurements

The Institute for Scientific Information (ISI) publishes the Social Sciences Citation Index Journal of Citation Reports (JCR). The JCR uses a variety of measures to extend the use of citation analysis to evaluate individual journals [4]. These measures are the citing and cited half-life, the immediacy index, and the impact factor. They characterise journals and conferences that publish proceedings.

The *citing half-life measure* is used to determine the age of the majority of articles referenced by the journal. It is the number of journal publication years, going back from the current, that accounts for at least 50% of the total citations made in it in the current year.

The *cited half-life measure* is used to determine the age of the majority of articles that cite a journal. It is the number of journal publication years, going back from the current, that accounts for at least 50% of the total citations made to it in the current year.

It is calculated by examining the chronological distribution of the publication years of works in which it is cited and determining how many years back one needs to go to cover at least 50% of these publications.

The *immediacy index* indicates how often journal articles are cited within the year of publication of the journal. It is the quotient of the number of citations that an article receives in a year and the total number of articles published in that year. When calculating the immediacy index for a journal, it becomes the quotient of the number of citations to all articles in the journal and the total number of of source items published in the given year.

The *impact factor* attempts to measure the relative importance of a journal in a particular field. It is used to evaluate performance over time. It is the quotient of the number of citations to published articles over a fixed period of time and the total number of articles published in that time period. Journal performance can be evaluated using these measures in order to aid purchasing decisions, or in an attempt to document objectively journal importance in academic evaluation.

3.4 Methods of Citation Analysis

A vast array of analytical methods for analysing citations exist. These can be grouped into two broad categories. Some methods simply count the number of citations to documents, authors, etc. Other methods examine the links between ideas or intellectual traditions. We will give a brief overview of some of the common methods of citation analysis.

3.4.1 Citation Counting

By counting citations, it is possible to determine how many times a document or set of documents has been cited by another document or sets of documents. Evaluative bibliometric analysis [40] uses citation counts for performance evaluation, i.e. it is used to evaluate and compare the performance of scientists, institutions, nations, and journals; often this results in rank ordering of the designated unit for the purpose of comparison. Garfield [17] pointed out that citation counting is not without limitations, especially when used to evaluate academic or research performance of individuals. The following factors are usually not taken into consideration when doing citation counts: the quality of the publication in which the work was published, the quality of the work that is citing, the context in which the citing is made, and the personal reasons for citing such as self-citation or citing the work of friends or colleagues. These factors, combined with a data set of insufficient size can cause results to be unreliable.

3.4.2 Bibliographic Coupling

Bibliographic coupling [23] defines coupling units. A coupling unit is a single reference cited by two or more documents. The number of coupling units shared by publications indicate the strength of the coupling between the documents. The static link between publications and their dated references produced by bibliographic coupling is unfortunate as the ever changing scientific landscape is presumed to produce dynamic literature.

3.4.3 Document Co-citation Analysis

Co-citation analysis [49] attempts to overcome the limitations of bibliographic coupling. Rather than measuring links between source documents, it focuses on cited documents. The unit of measure is defined as the number of source documents that cite a pair of reference documents. If a pair of reference documents are cited in a large number of source documents, then the pair is said to have a strong measure of co-citation. Co-citation strength is dynamic, as each time a new source document is published it can influence the co-citations strength of any number of pairs of cited documents.

3.4.4 Author Co-citation Analysis

Author co-citation analysis [36] investigates prominent authors that are cited together. A list of prominent authors who represent a wide variety of academic activity is drawn up. All possible prominent author pairs are then constructed. References are then examined for the occurrence of these author pairs. These results are analysed and a measure is calculated. Author co-citation is useful in fields where the number of published works is small. If an author changes focus though, it may be ignored by this measure as authors' impact over all past works is taken into account.

3.4.5 Co-word Analysis

Co-word analysis [46] is used to establish links between documents. A list of key words is constructed from words occurring in say the titles and/or abstracts of documents. Care must be taken to limit the number of terms

to keep the computational complexity to manageable levels. Documents are then analysed for the occurrence of word pairs from the list of key terms. Leydesdorff *et al.* [33] suggested that keywords taken from document titles produce better results than key words taken from abstracts. They also suggested that co-word analysis is more useful for classifying the academic specifics of documents that share conceptual links than exploratory studies.

3.5 Quantitative Laws of Bibliometrics

Three interesting observations that have been gleamed from citation analysis are summarised by Lotka's Law, Bradford's Law and Zipf's Law. These laws are not scientific facts, but they pass general comment on the productivity of authors and the quality of journals. Zipf's Law makes some interesting observations of word frequencies in general text.

3.5.1 Lotka's Law

Lotka's Law comments on the productivity of authors. It states that the number of authors making n contributions is about $1/n^2$ of those making one contribution. Furthermore, about 60% of authors only make a single contribution. Table 3.1 quantifies Lotka's Law for authors making 1, 2, 3, 4, and 5 contributions respectively.

Number of contributions	1	2	3	4	5
Percentage of authors	60	15	6.66	3.75	2.4

Table 3.1: Lotka's Law

These numbers are approximations used to illustrate general trends.

3.5.2 Bradford's Law

Bradford's Law comments on the prestige/quality of journals. It states that the journals in a field can be divided into three groups.

- A core group of a few journals produce about one third of all papers in any given field of research.

- A second, larger, group of journals collectively contain approximately the same number of articles as the journals in the first group.
- A third, much larger group of journals contain approximately the same number of articles as the other two groups of journals.

A relationship of $1 : n : n^2$ exists between the three groups. Bradford's Law is a guide and will not apply to all journals.

3.5.3 Zipf's Law

Zipf's Law comments on the frequency distribution of words in written text. It states that if words in a text are ranked in order of decreasing frequency, then the product of the frequency f of each word and the rank r of that word in the ordered list will be a constant k. That is, $f * r = k$. Zipf's Law relies on the text being of sufficient length and is only a general observation, not scientific fact.

3.6 Conclusion

Citation analysis is an established, well researched field. Big institutions base their decisions about awarding grants, tenure, organisational hiring, and other purposes on evaluative bibliometrics. We will use citation counting to identify papers that have made a significant contribution to a given field or are regarded as useful by other researchers in the given field. It is dynamic, computationally simple and free of subjective lists and influences. Garfield pointed out that it is not without controversy amongst scientists and academics. If used incorrectly, it can be used to discriminate unfairly against individuals. A large dataset constructed without subjective bias will however ensure that the potential problems arising from using citation counting is avoided. Bias will be avoided by using random selection and every effort will be made to ensure as large a dataset as possible. Furthermore, the use of other performance measures, such as citing and cited half-life, immediacy index, and impact factors will enable us to make objective comparisons of academic documents.

Chapter 4

Automatic Extraction of Citation Information

4.1 Introduction

4.1.1 Motivation

Authors and publishers alike are beginning to make scientific publications available on the WWW in increasing numbers. Once a niche market for some specialty companies who tediously kept track of publications and citations, the focused search for and automatic extraction of meta-information from the vast number of articles available on the world wide web now becomes feasible. Special-purpose search engines [30] and focused web crawlers which collect information such as citation indices [31] and related bibliometric information, e.g. the impact an individual article had on a scientific discipline, are beginning to emerge. We are interested in improving the accuracy with which information can be extracted from the bibliographies of scientific articles that are available on the WWW, in order to be used for bibliometric purposes. Recently, a heuristic for extracting bibliographic entries was proposed that first parses those fields that have relatively uniform syntax, position, and composition. In addition, it uses syntactic relationships between fields and dictionaries of author names and journal titles to help identify fields [31]. Our method uses hidden Markov models (HMMs)

which recognise the style in which a bibliography was written. It then identifies fields of a citation with high accuracy (up to 97%) *without* the need for dictionaries. In fact, the use of dictionaries for names, journal titles, and the English language has a negligible effect on the result. In addition, the system is modular, i.e. new bibliography styles can very easily be accommodated by training a new HMM and adding a lexical analyser for that new bibliography only.

4.1.2 Bibliography Styles

Bibliography styles define the order and form in which references appear in bibliographies These conventions strive to give a collection of articles such as journals and edited volumes a uniform look. Some styles differ from each other significantly while others only have minor differences. Consider a reference for the same article in IEEE, AAAI, and NEWAPA styles, respectively:

> [1] S. Lawrence, C.L. Giles, and K. Bollacker, "Digital libraries and autonomous citation indexing", *IEEE Computer*, vol. 6, no.4, pp. 67–71, 1999.

> [Lawrence, Giles, & Bollacker1999] Lawrence, S.; Giles, C.L.; and Bollacker, K. 1999. Digital libraries and autonomous citation indexing ing, *IEEE Computer*, 32(6): 67–71.

> [Lawrence et al.1999] Lawrence, S., Giles, C.L., and Bollacker, K. (1999). Digital libraries and autonomous citation indexing. *IEEE Computer*, *32* (**6**), 67–71.

Bibliometric parsing is feasible if an article consistently follows a particular bibliography style that is known *a priori*. When the style is *unknown* it is more difficult. The extraction process must be highly accurate if the extracted information is to be used for bibliometric purposes. For this thesis, we used LaTeX style files to produce bibliographies in IEEE, AAAI, and NEWAPA styles. We propose the use of hidden Markov models (HMMs) for building an intelligent bibliography extractor which reliably recognises different bibliography styles that may differ only slightly from each other, and extracts the individual fields, i.e. authors, title, publisher, year, etc.

4.2 Hidden Markov Models

4.2.1 Preliminaries

A hidden Markov model (HMM) describes a process which goes through a finite number of non-observable states whilst generating either a discrete or continuous observation. Traditionally, HMMs have been used for statistical pattern recognition and signal processing, e.g. speech recognition [44], topic spotting [53], and part-of speech tagging [25]. More recently, they have also been used for protein sequence analysis [21], and extraction of information from texts [5, 32]. These approaches typically use hand-crafted models assembled by inspecting training data [15]. A few approaches also learn the unknown model (e.g.[47]).

A hidden Markov model probabilistically links the observation to the state transitions in a system. Algorithms exist that

- calculate the probability $P(O|M)$ for a HMM M generating a particular observation sequence O, (*Forward* algorithm),
- identify the most likely state sequence a system went through in generating the observed signal through the (*Viterbi* algorithm), and
- iteratively re-estimate (Baum-Welch) the HMM parameters given an observation sequence as training data (*Baum-Welch* algorithm). These formulas maximise the probability of the sequence being generated by the model.

The term "hidden" refers to a process' state transition sequence which is hidden from the observer. The process reveals itself to the observer only through the generated observations. A HMM is parameterised through a matrix of transition probabilities between states and output probability distributions of observations given the internal process state. These probabilities are used in the above mentioned algorithms for achieving the desired results.

Two types of HMMs are: *Ergodic* HMMs allow transitions between any two states whereas *left-to-right* models only allow transitions from states to themselves and to their immediate right neighbors. The type of HMM used depends on the application. Often, left-to-right models incorporate a lot of

domain knowledge which can lead to better performance compared to more general ergodic models which contain less prior knowledge.

A bibliographical entry can be viewed as a sequence of fields, e.g. author names, title, publisher, year, etc. The order of fields varies for different bibliography styles. In order to reliably extract the fields from a bibliography of *unknown* type, we need to determine the bibliography model that most likely generated the bibliographic entry. HMMs provide a natural framework for modelling the production of bibliographies.

4.2.2 HMM Training

A HMM M is a 5-tuple $M = (Q, A, B, \Sigma, \Pi)$ where $Q = \{q_1, \ldots, q_N\}$ is a set of states, $A = \{a_{ij}\}$ is a matrix which defines the transition probabilities between states, $B = \{b_i\}$ are probability density functions describing the occurrence of symbols from an alphabet Σ in HMM state q_i, and $\Pi = \{\pi_i\}$ represents the initial state distribution. For continuous observations, O is a probability density function; for discrete observations, O can be represented as a histogram $O = \{o_{ij}\}$ which counts the occurrences of each symbol σ_i in a given state q_j. For unlabelled data, training a HMM consists of estimating the transition and observation probabilities. The Baum-Welch training algorithm maximises the likelihood $P(S|M)$ of a model M generating a sequence S by iteratively re-estimating the parameters of the HMM. In the case of discrete HMMs where labelled data is available, training is very simple: We can count the number of state transitions $c(q_i \rightarrow q_j)$ (from state q_i to q_j) and the number $c(q \uparrow \sigma_k)$ (occurrences of symbol σ_k in state q) of symbols occurring in each state. This allows us to derive the maximum likelihood estimates for the model parameters:

$$\hat{P}(q_i \rightarrow q_j) = \frac{c(q_i \rightarrow q_j)}{\sum_{q_i, q_j \in Q} c(q_i \rightarrow q_j)} \tag{1}$$

and

$$\hat{P}(q_i \uparrow \sigma_k) = \frac{c(q_i \uparrow \sigma_k)}{\sum_{\sigma_k \in \Sigma} c(q_i \uparrow \sigma_k)} \tag{2}$$

This completes the training process.

4.2.3 Bibliography Style Identification

Given a set of trained HMMs where each model describes the generation of a bibliography entry in a specific style, we wish to identify the model that is most likely to have generated the bibliography entry at hand. Instead of following all possible paths through all HMMs, we use the Viterbi algorithm to compute the most likely state sequence the models went through while generating the bibliography entry. We then choose the model with the highest likelihood to extract the individual fields of an entry.

The Viterbi algorithm recursively computes the most likely state sequence as follows:

1. *Initialisation*

$$\delta_1(i) = \pi_i b_i(O_1) \quad 1 \leq i \leq N \tag{3}$$

$$\psi_1(i) = 0 \tag{4}$$

2. *Recursion*

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] b_j(O_t) \quad 2 \leq t \leq T \quad 1 \leq j \leq N \tag{5}$$

$$\psi(j) = \arg \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \quad 2 \leq t \leq T \tag{6}$$

3. *Termination*

$$P^* = \max_{1 <= i <= N} [\delta_T(i)] \tag{7}$$

$$q_T^* = \arg \max_{1 <= i <= N} [\delta_T(i)] \tag{8}$$

4. *Path (state sequence) backtracking*

$$q_t^* = \psi_t(q_{t+1}^*) \tag{9}$$

For this application, this formulation of the Viterbi algorithm can easily lead to underflow as the probabilities can become very small. Thus, we used logarithmic scaling in order to prevent this numerical instability. The equations for the Viterbi algorithm then become:

1. *Initialisation*

$$\delta_1(i) = log(\pi_i) + log(b_i(O_1)) \tag{10}$$

2. *Recursion*

$$\delta_t(j) = \max_{1<=i<=N}[\delta_{t-1}(i)log(a_{ij})] + log(b_j(O_t)) \quad (11)$$

3. *Termination*

$$log(P^*) = \max_{1<=i<=N}[\delta_T(i)] \quad (12)$$

The other equations remained unchanged.

4.3 Bibliography Extraction

4.3.1 Preprocessing

Generally, obtaining labelled training data is tedious and expensive as the data labeling has to be done manually. Our application requires that the fields in bibliographies be labelled. The use of LaTeX bibliography styles alleviates the problem somewhat as we use BibTeX bibliographies which identify the individual fields; bibliographies can then be generated automatically in any style desired.

Before data is fed to the system it is first converted to tokens by a lexical analyser. This allows words and symbols documents in the data to be mapped to 32 tokens:

```
TOKENS={
        <OBRACKET>, <NUMBER>, <CBRACKET>, <CAPITAL>,
        <PINITIAL>, <UWORD>, <TERM>, <LWORD>, <COLON>,
        <DASH>, <ORBRACKET>, <CRBRACKET>, <SLASH>, <QMARK>,
        <DQUOTE>, <SQUOTE>, <BQUOTE>, <PLUS>, <KAPPIE>,
        <TILDE>, <ASTRIX>, <FSTOP>, <SCOLON>, <COMMA>,
        <PYEAR>, <BSLASH>, <DOLLAR>, <NAME>, <AND>,
        <WORD>, <AMPLISAND>, <MONTH>
       }
```

It is much simpler for the model to deal with these tokens than all the words and symbols in the original text.

4.3.2 Labelled vs. Unlabelled Data

It is possible to randomly initialise a HMM and train it on unlabelled data only; however, labelled data is valuable as it enables us to derive the maximum likelihood estimates for the parameters of the HMM. In order to generate the labelled data consisting of tags for all fields in the individual bibliography entries, we used a tool for manipulating BibTeX bibliography files; it produces a bibliography in a desired style. We need to identify the states in order to model the problem. Probabilities are then calculated for the token distribution within states and the transitions between states. There are two popular methods or performing this task: If labelled data is available, it can be used to initialise the state and transition probabilities. If labelled data is not available the state and transition probabilities can be randomly initialised and then fine-tuned using a training algorithm, e.g. Baum-Welch. If a limited amount of labelled data is, available the above two methods can be used together to set up the model.

4.4 Experiments

4.4.1 Basic Model

We constructed basic models for classification and extraction purposes. These models consist of states for each of the fields as identified by the bibliography style and the token set listed above.

Seen Data

We hand-crafted HMMs for the three bibliography styles IEEE, AAAI, and NEWAPA. The latter two styles are quite similar to each other, but are significantly different from the IEEE style. In order to check that the model learned, we trained the HMMs on 1212 bibliography entries for each style and then used the same data to test the models. The training performance for each style is shown in Table 4.1. We observe that the fields are correctly extracted from the bibliographic entries with high probability; the model recognises the input strings with similar log probabilities $P(B|M_i)$, which will be used when trying to recognise the style used to format the input.

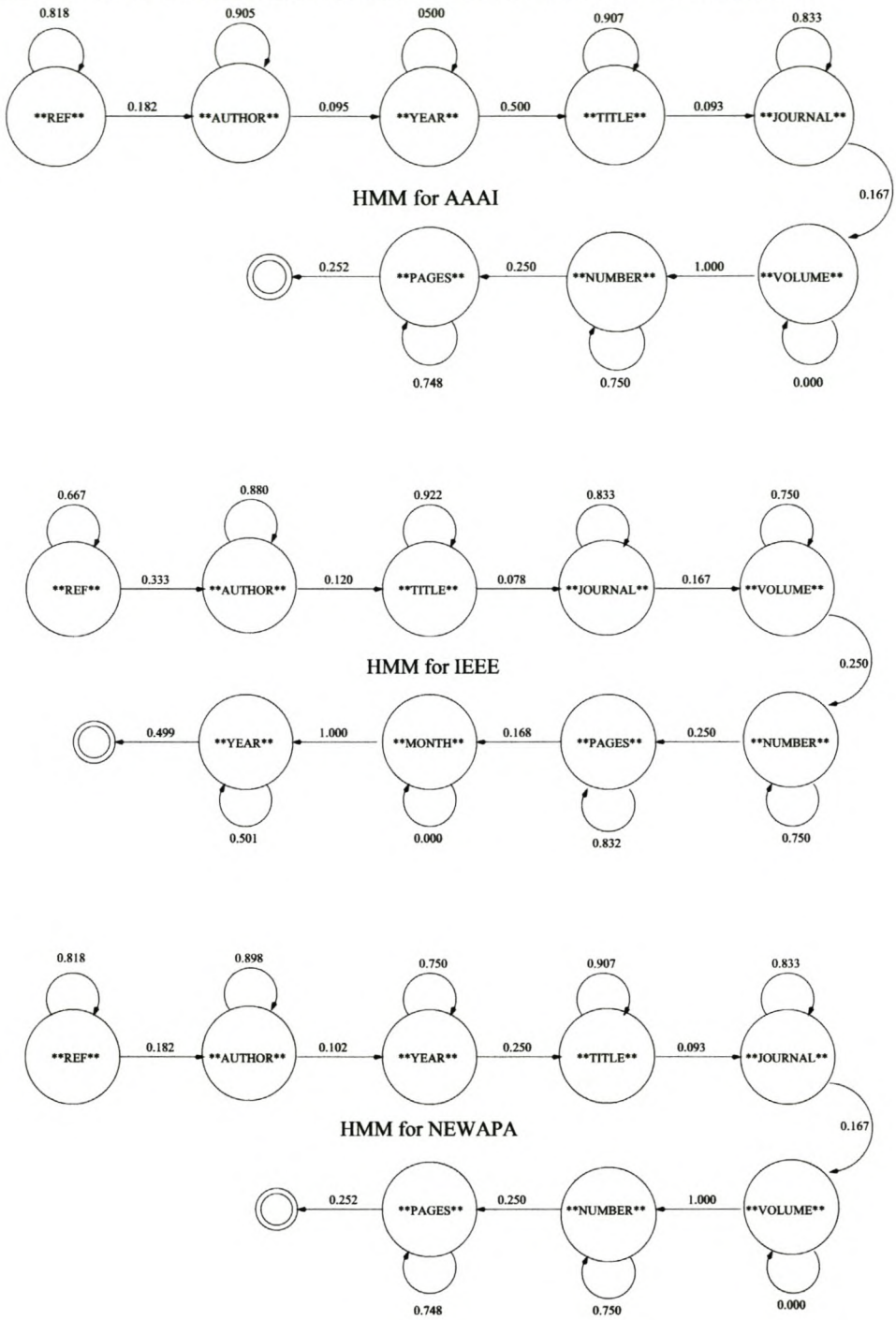


Figure 4.1: **Trained Hidden Markov Models:** These are the models trained to recognise IEEE, AAAI, and NEWAPA bibliography styles.

HMM Training	AAAI	IEEE	NEWAPA
Entries Train	1212	1212	1212
Entries Test	1212	1212	1212
Tokens	54617	57538	57167
Correct Tokens	94.9%	84.6%	97.8%
log Probability	-77.85	-81.90	-81.56
Valid Entries	1212	1212	1212
Tokens per Entry	55	58	57

Table 4.1: **Training Performance:** The table shows for each bibliography style the number of entries used to train HMMs, the number of entries used to test the HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability $P(B|M_i)$ of a particular model M_i having generate a bibliographic entry B, the number of valid entries, and the average number of tokens per entry.

Unseen Data

We trained the three HMMs on only 212 bibliographic entries generated in the respective styles and tested the recognition and field identification performance for each of the three styles on the remaining 1000 entries. The HMMs with transition probabilities are shown in Figure 4.1. The results in Table 4.2 show the parsing performance of the HMMs for the three bibliography styles when presented with entries generated by the corresponding styles. Even though only a small subset of all examples were used for training, the HMMs extract the fields from the bibliographic entries with high accuracy. The results in Table 4.3 illustrate the parsing performance of the HMMs for the three bibliography styles when presented with bibliographies generated using a different style. Even though the AAAI and NEWAPA styles are quite similar, the number of valid entries recognised by a HMM when presented with an entry of a different style is very low. This demonstrates that there exists almost no functional overlap between parsers for different styles, i.e. the likelihood of fields in an entry being correctly identified by the wrong parser is very low.

Finally, the results in Table 4.4 demonstrate the ability of the three HMMs to recognise the different bibliography styles. We tested each of the HMMs on bibliographies generated using each of the three styles IEEE, AAAI, and NEWAPA. The model M_i with the highest average logarithmic probability

HMM Testing	AAAI	IEEE	NEWAPA
Entries Train	212	212	212
Entries Test	1000	1000	1000
Tokens	45188	47560	47290
Tokens Wrong	2650	7558	1424
Correct Tokens	94.1%	84.2%	97.0%
log Probability	-77.53	-81.71	-81.11
Valid Entries	977	988	976
Tokens per Entry	45	48	47

Table 4.2: **Testing Performance:** The table shows for each bibliography style the number of entries used to train the HMMs, the number of entries used to test the trained HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability of a particular model having generated a bibliographic entry, the number of valid entries, and the average number of tokens per entry.

HMM Cross Testing	AAAI	IEEE	NEWAPA
Style	NEWAPA	AAAI	AAAI
Entries	1000	1000	1000
Tokens	47290	45188	45188
log Probability	$-\infty$	$-\infty$	-99.29
Valid Entries	0	0	177
Tokens per Entry	47	45	45

Table 4.3: **Cross-Testing Performance:** The table shows for each bibliography style the number of entries used to cross-test the trained HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability $logP(B|M_i)$ of a particular model M_i having generated a bibliographic entry B, the number of valid entries, and the average number of tokens per entry.

Data	Model		
	AAAI	IEEE	NEWAPA
AAAI	-77.53	$-\infty$	-99.29
IEEE	$-\infty$	-81.71	$-\infty$
NEWAPA	-∞	$-\infty$	-81.11

Table 4.4: **Discrimination Performance:** The table shows the average logarithmic probability $P(B|M_i)$ of each HMM M_i having generated a bibliography B in all three styles. The largest value for $P(B|M_i)$ identifies the bibliography style.

$P(B|M_i)$ identifies the style used to generate a particular bibliography B[1]. Clearly, the HMMs reliably identify the bibliography styles.

4.4.2 Boundary States

Punctuation lends structure to written text. It makes it easier to read and understand text by giving the reader additional information about the text. It supplies information such as start and end of sentences, whether a sentence is a question or not, where to pause and many other pieces of information that make the text more readable. Bibliography styles also make use of punctuation and other symbols to structure entries. This structure forms a skeleton for the bibliography style. Punctuation occurs in both the sentences contained in the bibliography entries, as well as the skeleton. If this was not the case then it would have been simple to extract each field relying only the punctuation used in the skeleton. However, this is not the case and distinguishing between between the punctuation used in the fields and between the fields is not trivial. This section investigates the use of this knowledge to try and determine the edges of the entry fields. We introduce boundary states that only accept a very limited set of symbols. The objective is to bias the probabilities of the paths in such a way that very high probabilities are assigned to the correct paths, and very low probabilities to the incorrect paths. There is also the additional benefit that the parsed fields will be

[1]Other methods for computing the score of a bibliography are possible. The *log-odds ratio* $P(B|M)/P(B|0)$ where $P(B|0)$ is the probability that a bibliography B was generated by a *null model* whose parameters reflect the general bibliography distribution on the training set. Alternatively, one can adapt the Milosavljevic algorithmic significance test developed for DNA sequence analysis. It defines a threshold $t = -log(\sigma) + log(N)$ where N is the number of bibliographic entries, and σ is a significance level.

HMM Training	AAAI	IEEE	NEWAPA
Entries Train	1212	1212	1212
Entries Test	1212	1212	1212
Tokens	54617	57538	57167
Correct Tokens	100.0%	99.9%	100.0%
log Probability	-57.80	-58.97	-58.74
Valid Entries	1212	1212	1212
Tokens per Entry	55	58	57

Table 4.5: **Training Performance:** The table shows for each bibliography style the number of entries used to train HMMs, the number of entries used to test the HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability $P(B|M_i)$ of a particular model M_i having generated a bibliographic entry B, the number of valid entries, and the average number of tokens per entry.

free of unwanted punctuation marks and other symbols used to demarcate boundaries.

Seen Data

Again the HMMs were trained on 1212 bibliography entries for each style and then tested with the same data, to see if the model learned. The training performance for each style is shown in Table 4.5. We observe that the fields are correctly extracted from the bibliographic entries with much higher probability; the values of the log probabilities $P(B|M_i)$ have also improved.

Unseen Data

We again trained the three HMMs on only 212 bibliographic entries generated in the respective styles and tested the recognition and field identification performance for each of the three styles on the remaining 1000 entries. The HMMs with transition probabilities are shown in Figure 4.2.

The results in Table 4.6 show the parsing performance of the HMMs for the three bibliography styles when presented with entries generated by the corresponding styles that now include the boundary states. The percentage of correctly classified tokens has improved to 99%. We also observe that the models' ability to discriminate between the different styles has improved (see also Tables 4.4 and 4.8).

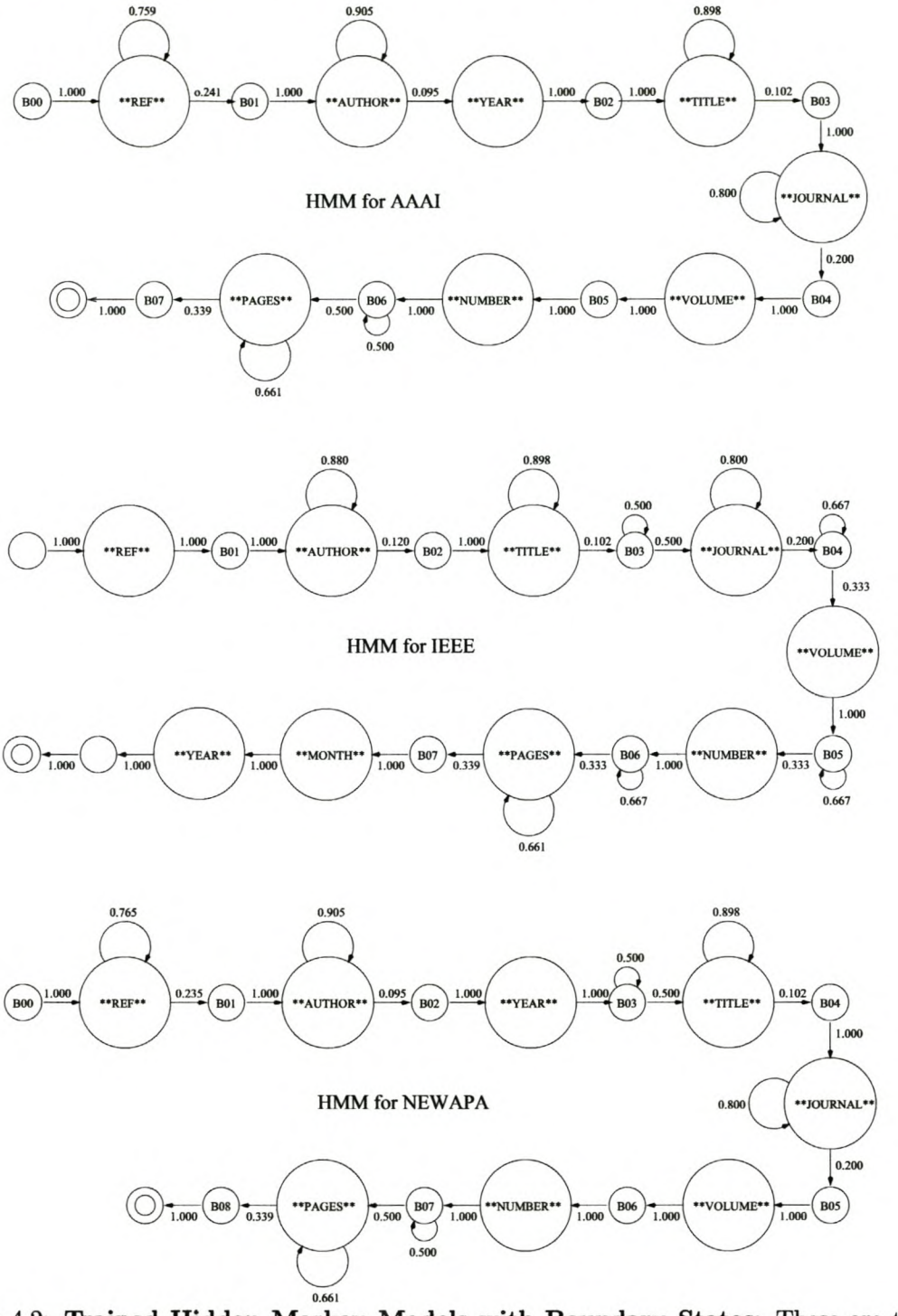


Figure 4.2: **Trained Hidden Markov Models with Boundary States:** These are the models, with boundary states, trained to recognise IEEE, AAAI, and NEWAPA bibliography styles.

HMM Testing	AAAI	IEEE	NEWAPA
Entries Train	212	212	212
Entries Test	1000	1000	1000
Tokens	45188	47560	47290
Tokens Wrong	387	470	449
Correct Tokens	99.1%	99.0%	99.1%
log Probability	-57.45	-58.75	-58.34
Valid Entries	978	982	977
Tokens per Entry	45	48	47

Table 4.6: **Testing Performance:** The table shows for each bibliography style the number of entries used to train the HMMs, the number of entries used to test the trained HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability of a particular model having generated a bibliographic entry, the number of valid entries, and the average number of tokens per entry.

The results in Table 4.7 illustrate the parsing performance of the HMMs with boundary states for the three bibliography styles when presented with bibliographies generated using a different style.

Even though the AAAI and NEWAPA styles are quite similar, the number of valid entries recognised by a HMM when presented with an entry of a different style is zero. This demonstrates that there now exists no functional overlap between parsers for different styles, i.e. the likelihood of fields in an entry being correctly identified by the wrong parser is very low.

Tables 4.2 and 4.6 showed the performance of the models by examining the overall token classification for HMMs with and without boundary states, respectively. The introduction of boundary states improved the token classification performance on unseen data for all three styles. The number of misclassified tokens per entry improved more significantly, as shown in Table 4.9. With boundary states, we were able to correctly extract all fields for more than 97% of all entries. This is a significant improvement compared to the performance of HMM's without boundary states. However, when table 4.9 is examined, it is observed that the decrease in the number of errors per entry is very significant. Entries are either accepted correctly or not at all. Table 4.6 show the results of three experiments, one for each of the three bibliographic styles. The experiment was however repeated 30 times for each of the bibliographic styles. This enables us to construct table 4.10. From

HMM Cross Testing	AAAI	IEEE	NEWAPA
Style	NEWAPA	AAAI	AAAI
Entries	1000	1000	1000
Tokens	47290	45188	45188
log Probability	$-\infty$	$-\infty$	$-\infty$
Valid Entries	0	0	0
Tokens per Entry	47	45	45

Table 4.7: **Cross-Testing Performance:** The table shows for each bibliography style the number of entries used to cross-test the trained HMMs, the number of tokens generated, the percentage of correct token classification, the average log probability $logP(B|M_i)$ of a particular model M_i having generated a bibliographic entry B, the number of valid entries, and the average number of tokens per entry.

Data	Model		
	AAAI	IEEE	NEWAPA
AAAI	-57.45	$-\infty$	$-\infty$
IEEE	$-\infty$	-58.75	$-\infty$
NEWAPA	$-\infty$	$-\infty$	-58.34

Table 4.8: **Discrimination Performance:** The table shows the average logarithmic probability $P(B|M_i)$ of each HMM M_i having generated a bibliography B in all three styles. The largest value for $P(B|M_i)$ identifies the bibliography style.

	AAAI		IEEE		NEWAPA	
Boundary States	No	Yes	No	Yes	No	Yes
0 errors	0	978	0	982	0	977
1 error	0	0	0	0	958	0
2 errors	895	0	0	0	12	0
3 errors	22	0	0	0	5	0
4 errors	17	0	46	0	1	0
5 errors	15	0	97	0	1	0
>5 errors	51	22	857	18	23	23

Table 4.9: **Token Classification:** This table shows the number of tokens classified incorrectly for each entry.

Measure	Model		
	AAAI	IEEE	NEWAPA
Average Performance	97.7%	95.6%	98.7%
Standard Deviation	1.59	3.24	0.9
Upper Bound	98.3%	96.8%	99.0%
Lower Bound	97.1%	94.4%	98.3%

Table 4.10: **Confidence in Results:** This table shows the Average, Standard Deviation, and Confidence Interval for each of the three models. The results from 30 experiments are use. The confidence level is set at 95%.

these results we can see that the models will perform in the specified ranges 95% of the time.

4.5 Conclusions

We have applied hidden Markov models (HMMs) to the problem of extracting the fields of bibliographic entries *without* a priori knowledge of the bibliography style used to generated the entries. We trained HMMs on a small subset of bibliographies generated using the IEEE, AAAI, and NEWAPA styles; we then tested the models on a larger set of previously unseen data. We observed that the log probability $logP(B|M_i)$ of HMM M_i having generated a given bibliography B reliably identifies the particular style used for generating a bibliography. Furthermore, the models extract the fields of bibliographic entries when generated by the corresponding bibliography styles with very high accuracy, up to 97%. The approach is modular, i.e. new HMMs and parsers can easily be added as new styles are introduced.

Chapter 5

Classification of Scientific Literature on the WWW

5.1 Introduction

There are several ways the above concepts can be used to classify scientific data found on the WWW. One can construct a search engine that searches the WWW for scientific papers about a given topic. The relevant bibliography information can then be extracted from these papers and used to rank them. If one considers the size of postscript files, even with compression, can easily exceed one megabyte it is clear that the volume of traffic generated exceeds the gains made. Alternatively, a web crawler can be constructed that can crawl the WWW gathering papers about given topics or general subject areas. These papers can then be archived. Searches can be performed on this archive and the extraction and citation analysis techniques used to rank the papers.

The rest of this chapter describes how we used HMMs to extract bibliometric information from academic documents and then used this extracted information to determine the works that the researchers regarded as the most useful. We performed this task on the following three topics: Hidden Markov Models (HMMs), Neural Networks (NNs) and Recurrent Neural Networks (RNNs). The first was chosen as HMMs are used in the extraction process. The latter were chosen to see how big the overlap/distinction is between two fields that may seem similar at first glance.

5.2 Special Considerations

Postscript (ps) and Portable Document Format (pdf) are two of the most popular formats used for publishing papers on the WWW. We downloaded documents from Citeseer and converted them text. We extracted bibliometric information from these documents. There are a lot of bibliography styles outside of the three described in Chapter 4. We had to construct additional models to cover all the formats. As described in Chapter 4, this construction of additional models is a straightforward process. Once we identified a new style, a new model was constructed and trained. Some variations within the styles were not well represented in the initial training set, e.g. there were no references that contained URLs. Rather than adding URLs to the training set, we constructed additional models to handle these variations. This results in several models for the same referencing style, but for different variations within that style, e.g. one model for NEWAPA articles and a different model for NEWAPA URLs. In Chapter 4, we showed how well the models discriminate between subtle variations in format, so this posed no problems.

We use the extracted information to find out how many times each paper was cited by the downloaded works, i.e. we had to keep track of all the extracted references. All punctuation is removed from the extracted information and all text is converted to lower case. We use the title, authors and year to identify each reference. We use no stemming or stop words. Each reference has a counter associated with it. Each time a duplicate is found the counter is increased.

5.3 Case Study: Computer Science Literature

5.3.1 Hidden Markov Models (HMMs)

We performed searches using the keywords *hidden Markov model*, *hidden Markov models* and *hmm* on the site *http://citeseer.nj.nec.com/cs*. We downloaded 82 documents. We successfully converted 72 documents to text. These documents contained a total of 1837 references. Information was extracted from 1743 of these references. These results are summarised in Table

5.1.

Papers Downloaded From Citeseer	82
Papers Successfully Converted To Text	72
Number Of References Contained In Converted Documents	1837
Number Of References From Which Information Was Extracted	1742

Table 5.1: Reference summary for HMMs

This information is then used to determine how many times each of the papers referenced within the downloaded papers, is referenced. These results are summarised in Table 5.2.

1412	Unique References
86.90 %	Of Papers Cited Only Once
30.56 %	Of Papers Referenced The Most Cited Paper
25.51	References Per Paper

Table 5.2: Extracted information summary for HMMs

The most cited papers are listed below:

1. Maximum Likelihood From Incomplete Data Via The EM Algorithm, A. P. Dempster, N. M. Laird, and D. B. Rubin, 1977
2. A Tutorial On Hidden Markov Models And Selected Applications In Speech Recognition, L. R. Rabiner, 1989
3. An Introduction To Hidden Markov Models, L. R. Rabiner and B. H. Juang, 1986
4. Probabilistic Independence Networks For Hidden Markov Probability Models, P. Smyth, D. Heckerman, and M. Jordan, 1997
5. Probabilistic Reasoning In Intelligent Systems: Networks Of Plausible Inference, J. Pearl, 1988
6. Hidden Markov Models For Speech Recognition, X. D. Huang, Y. Ariki, and M. A. Jack, 1990

7. Pattern Classification And Scene Analysis, R. O. Duda and P. E. Hart, 1973
8. Hidden Markov Models In Computational Biology: Applications To Protein Modeling, A. Krogh, M. Brown, I. S. Mian, K. Sjolander, and D. Haussler, 1993
9. Mean Field Networks That Learn To Discriminate Temporally Distorted Strings, C. K. I. Williams an G. E. Hinton, 1991
10. Hierarchical Mixtures Of Experts And The EM Algorithm, M. I. Jordan, and R. A. Jacobs, 1994
11. Hidden Markov Models Of Biological Primary Sequence Information, P. Baldi, Y. Chauvin, T. Hunkapillar and M. McClure, 1994
12. A Maximization Technique Occurring In The Statistical Analysis Of Probabilistic Functions Of Markov Chains, L. E. Baum, T. Petrie, G. Soules, N. Weiss, 1970

5.3.2 Neural Networks (NNs)

We performed searches using the keywords *neural network* and *neural networks* on the site *http://citeseer.nj.nec.com/cs* and downloaded 117 documents. We successfully converted 74 of the documents to text. These documents contained a total of 2015 references. Information was extracted from 1894 of these references. These results are summarised in Table 5.3.

Papers Downloaded From Citeseer	117
Papers Successfully Converted To Text	74
Number Of References Contained In Converted Documents	2015
Number Of References From Which Information Was Extracted	1894

Table 5.3: Reference summary for NNs

This information was then used to determine how many times each of the papers referenced within the downloaded papers was referenced. These results are summarised in Table 5.4.

1699	Unique References
94.06 %	Of Papers Cited Only Once
10.81 %	Of Papers Referenced The Most Cited Paper
27.23	References Per Paper

Table 5.4: Extracted information summary for NNs

The most cited papers are listed below:

1. Learning Internal Representations By Error Propagation, D.E. Rumelhart, G. E. Hinton and R. J. Williams, 1986
2. Introduction To The Theory Of Neural Computation, J. Hertz, A. Krogh and R. Palmer, 1991
3. Hierarchical Mixtures Of Experts And The EM Algorithm, M. I. Jordan and R. A. Jacobs, 1994
4. Neural Networks For Pattern Recognition, C. M. Bishop, 1995
5. Maximum Likelihood From Incomplete Data Via The EM Algortithm, A. P. Dempster, N. M. Laird and D. B. Rubin, 1977
6. Neural Networks For Optimization And Signal Processing, A. Cichocki and R. Unbehauen, 1993
7. The Self-Organizing Map, T. Kohonen, 1990

5.3.3 Recurrent Neural Networks (RNNs)

We searched for scientific documents using the keywords *recurrent neural network* and *recurrent neural networks* on the site *http://citeseer.nj.nec.com/cs*. We downloaded 125 papers and successfully converted 96 of them to text. These documents contained a total of 2082 references. Information was extracted from 2051 of these references. These results are summarised in Table 5.5.

This information was then used to determine how many times each of the papers referenced within the downloaded papers was referenced. These results are summarised in Table 5.6.

Papers Downloaded From Citeseer	125
Papers Successfully Converted To Text	96
Number Of References Contained In Converted Documents	2082
Number Of References From Which Information Was Extracted	2051

Table 5.5: Reference summary for RNNs

1496	Unique References
82.42 %	Of Papers Cited Only Once
31.25 %	Of Papers Referenced The Most Cited Paper
21.69	References Per Paper

Table 5.6: Extracted information summary for RNNs

The most cited papers are listed below:

1. Finding Structure In Time, J. Elman, 1990
2. A Learning Algorithm For Continually Running Fully Recurrent Neural Networks, R. J. Williams and D. Zipser, 1989
3. Learning And Extracting Finite State Automata With Second-Order Recurrent Neural Networks, C. L. Giles, C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun and Y. C. Lee, 1992
4. Introduction To The Theory Of Neural Computation, J. Hertz, A. Krogh and R. G. Palmer, 1991
5. The Induction Of Dynamical Recognizers, J. B. Pollack, 1991
6. Learning Internal Representations By Error Propagation, D. E. Rumelhart, G. E. Hinton and R. J. Williams, 1986
7. Finite State Automata And Simple Recurrent Networks, A. Cleeremans, D. Servan-Schreiber, and J. L. McClelland, 1989
8. Learning Long-Term Dependencies With Gradient Descent Is Difficult, Y. Bengio, P. Simard and P. Frasconi, 1994
9. Induction Of Finite-State Languages Using Second-Order Recurrent Networks, R. L. Watrous and G. M. Kuhn, 1992

10. Learning State Space Trajectories In Recurrent Neural Networks, B. A. Pearlmutter, 1989
11. Introduction To Automata Theory, Languages, And Computation, J. E. Hopcroft and J. D. Ullman, 1979[1]
12. Hierarchical Mixtures Of Experts And The EM Algorithm, M. Jordan and R. Jacobs, 1994

5.4 Interpreting the results

We discussed in Chapter 3 the different reasons why people cite other works. A common theme that emerged from this discussion is that people cite works that are somehow related to their own work. Therefore, when authors cite work, they are acknowledging this link. At the same time they are suggesting further sources of information to the reader. We thus infer that authors are likely to cite works that they found useful, i.e. they have been recommended by researchers in the particular fields as being the most useful. We do not feel that the older works have an unfair advantage with this method. If someone was to publish a newer paper about a topic already covered by a previous work and the new paper presented the work in a more accessible way, we feel confident that it will replace the older work as the main source of inspiration. If authors prefer to continue citing an older work, then we feel it deserves its place on this list.

All three topics examined fall under the umbrella of machine learning. It is therefore not surprising to see some overlap between all three fields investigated. As expected there, there was even more overlap between NNs and RNNs. It is also interesting to note that while with HMMs and RNNs 87% and 82% respectively of papers were cited only once, with NNs 94% of paper were cited only once. One possible explination of this is the fact that CiteSeer indexes Computer Science papers and that NNs are also found in fields outside of computer science. A lot of the references extracted from the NN documents were to documents that would not be classified as computer science.

[1]This reference appears in this list of most cited documents because formal languages, and particularly deterministic finite state automata, have been found useful as a paradigm for studying issues of symbolic knowledge representation in, and extraction from, RNNs

5.5 Conclusion

We successfully extracted bibliometric information from scientific publications with very high accuracy. We showed that hidden Markov models are an appropriate tool for this extraction. The system was able to both identify which model to use and to then extract the information with a high degree of accuracy. We were able to use the extracted information to determine which sources were cited most often by authors in the respective fields.

Chapter 6

Conclusions and Directions for Future Research

6.1 Conclusions

The Internet is becoming an ever increasingly important source of information. As more academic works find their way onto the WWW, this trend will probably speed up even more. Unfortunately, as more information becomes available, it becomes increasingly difficult to find useful information. Fortunately, there are researchers around the world working hard to develop tools that will simplify the task of finding useful information on the WWW. In this thesis, we explored some of the study areas that specialise in improving our ability to find information on the WWW. We examined web crawlers, both focused and exhaustive, and how important they are in the process of gathering information. We investigated how focused crawlers can be used to minimise overhead when we are only looking for information about very specific topics. We briefly touched on how search engines are constructed on top of the indices that are built from the information gathered by the crawlers. This Internet specific knowledge can be combined with knowledge from seemingly unrelated fields, such as bibliometrics, to build powerful tools that can be used to greatly simplify the lives of researcher. This thesis shows how citation analysis can be combined with machine learning to work in conjunction with existing Internet tools. We successfully extracted information about works that are continuing to play an important part in shaping three

different fields of research.

6.2 Future Work

The overhead involved in building and maintaining a system like CiteSeer means that is probably not feasible to construct a similar system while there is a free, working and well maintained system already available. It is therefore unlikely that this system will ever be developed into a stand-alone system such as CiteSeer. It would be possible to integrate this system with an existing system such as Citeseer. It is far more likely, though, that the techniques used in this thesis can be applied to other areas of information extraction. The techniques discussed here can be used to extract information from any loosely structured data, such as conference announcements, home pages and catalogs. It is therefore suggested that this line of exploration is continued.

Bibliography

[1] Internet pioneers. *http://www.ibiblio.org/pioneers/cerf.html.*

[2] Nua internet how many online. *http://www.nua.ie/surveys/how_many_online/world.html.*

[3] Search engines statistics: Database total size estimates. *http://www.searchengineshowdown.com/stats/sizeest.shtml.*

[4] D. R. Baker. Citation analysis: A methodoligical review. *Social Work Research and Abstracts*, 26(3):3–10, 1990.

[5] D. M. Bikel, S. Miller, R. Schwartz, and Weischedel. Nymble: A high-performance learning name finder. In *Proceedings of ANLP-97*, pages 194–201, 1997.

[6] K. Bollacker, S. Lawrence, and C. L. Giles. CiteSeer: An autonomous web agent for automatic retrieval and identification of interesting publications. In Katia P. Sycara and Michael Wooldridge, editors, *Proceedings of the Second International Conference on Autonomous Agents*, pages 116–123, New York, 1998. ACM Press.

[7] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[8] R. D. Cameron. A universal citation database as a catalyst for reform in scholarly communication. *First Monday*, 1997.

[9] E. G. Coffman, Z. Liu, and R. R. Weber. Optimal robot scheduling for web search engines. Technical Report RR-3317, Bell Labs, Lucent Technologies.

[10] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *26th International Conference on*

Very Large Databases, VLDB 2000, pages 527–534, Cairo, Egypt, 10–14 September 2000.

[11] J. Donoghue. *Understanding scientific literatures: A bibliometric approach.* MIT Press, Cambridge, MA, 1973.

[12] R. B. Doorenbos, O. Etzioni, and D. S. Weld. A scalable comparison-shopping agent for the world-wide web. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 39–48, Marina del Rey, CA, USA, 1997. ACM Press.

[13] O. Etzioni. Moving up the information food chain: Deploying softbots on the world wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 1322–1326, Menlo Park, 4–8 1996. AAAI Press / MIT Press.

[14] O. Etzioni. The World-Wide Web: Quagmire or Gold Mine? *Communications of the ACM*, 39(2):65–68, 1999.

[15] D. Freitag and A. McCallum. Information extraction with hidden Markov models and shrinking. In *Proceedings of the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999.

[16] E. Garfield. Citation indexes for science. *Science*, 122:108–111, 1955.

[17] E. Garfield. Is citation analysis a legitimate evaluation tool? *Scientometrics*, 1(4):398–375, 1979.

[18] E. Garfield. Announcing the SCI compact disc edition: CD-ROM gigabyte storage technology, novel software, and bibliographic coupling make desktop research and discovery a reality. *Current Contents*, pages 3–13, 1988.

[19] C. Lee Giles, Kurt Bollacker, and Steve Lawrence. CiteSeer: An automatic citation indexing system. In Ian Witten, Rob Akscyn, and Frank M. Shipman III, editors, *Digital Libraries 98 - The Third ACM Conference on Digital Libraries*, pages 89–98, Pittsburgh, PA, June 23–26 1998. ACM Press.

[20] K. Hammond, R. Burke, C. Martin, and S. Lytinen. Faq finder: A case-based approch to knowledge navigation. In *Working Notes*

of the AAAI Spring Symposium on Information Gathering Heterogeneous, Distrubuted Environments, Distributed Environments, pages 69–73, Stanford University, 1995. AAAI Press.

[21] R. Hughey and A. Krogh. Hidden Markov models for sequence analysis: Extension and analysis of the basic method. *Computer Applications in the Biosciences*, 12:95–107, 96.

[22] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, pages 237–285, 1996.

[23] M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10–25, 1963.

[24] C. Knoblock and A. Levy, editors. *Working Notes of the AAAI Spring Symposium on Information Gathering Heterogeneous, Distributed Environments.* Number SS-95-08 in AAAI Spring Symposium Series. AAAI Press/MIT Press, 1995.

[25] J. Kupiec. Robust part-of-speech tagging using a hidden Markov model. *Computer, Speech and Language*, 6:225–242, 1992.

[26] S. Lawrence. Online or invisible? *Nature*, 411:521, 2001.

[27] S. Lawrence and C. L. Giles. Searching the world wide web. *Nature*, 280:98–100, 1998.

[28] S. Lawrence and C. L. Giles. Accessibility of information on the web. *Nature*, 400:107–109, 1999.

[29] S. Lawrence and C. L. Giles. Searching the web: General and scientific information access. *IEEE Communications*, 37(1):116–122, 1999.

[30] S. Lawrence and C.L. Giles. Context and page analysis for improved web search. *IEEE Internet Computing*, 2(4):38–46, 1998.

[31] S. Lawrence, C.L. Giles, and K. Bollacker. Digital libraries and autonomous citation indexing. *IEEE Computer*, 32(6):67–71, 1999.

[32] T. R. Leek. Information extraction using a hidden Markov model. Master's thesis, UC San Diego, 1997.

[33] L. Leydesdorf and R. Zaal. Co-words and citations relations between document sets and environments. In L. Egghe and R. Rousseau, editors, *Infometrics 87/88*, pages 105–119, New York, 1988. Elsevier.

[34] L. Leydesdorff and O. Amsterdamska. Dimensions of citation analysis. *Science, Technology and Human Values*, 15(3):305–335, 1990.

[35] O. A. McBrayn. Genvl and wwww: Tools for taming the web. In *Proceedings of the 1st Annual International World Wide Web Conference*, pages 4–11, 1994.

[36] K. McCain. Cocited author mapping as a valid representation of intellectual structure. *Journal of the American Society for Information Science*, 37(3):111–122, 1986.

[37] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[38] F. Menczer. ARACHNID: Adaptive retrieval agents choosing heuristic neighborhoods for information discovery. In *Machine Learning: Proceedings of the Fourteenth International Conference*, pages 227–235, 1997.

[39] W. Meng, C. T. Yu, and K.-L. Liu. Building efficient and effective metasearch engines. *ACM Computing Surveys*, 34(1):48–89, 2002.

[40] F. Narin. *Evaluative bibliometrics: The use of publiction and citation analysis in the evaluation of scientific activity.* U.S. Department of Commerce, Springfield, VA, 1976.

[41] K. Nigam, A. K. McCallum, S. Thrun, and T. M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[42] M. Perkowitz and O. Etzioni. Category translation: Learning to understand information on the internet. In *International Joint Conference on Artificial Intelligence, IJCAI-95*, pages 930–938, Montreal, Canada, 1995.

[43] Oxford University Press. *Oxford Dictionary of English.* Oxford University Press, Oxford, UK, 1999.

[44] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–285, 1989.

[45] J. Rennie and A. K. McCallum. Using reinforcement learning to spider the Web efficiently. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*,

pages 335–343, Bled, SL, 1999. Morgan Kaufmann Publishers, San Francisco, US.

[46] A. Rip and J. Courtial. Co-word maps of biotechnology: An example of cognitive scientometrics. *Scientometrics*, 6(6):381–400, 1984.

[47] K. Seymore, A. McCallum, and R. Rosenfeld. Learning hidden Markov model structure for information extraction. In *AAAI Workshop on Machine Learning for Information Extraction*, 1999.

[48] W. R. Shadish, D. Tolliver, M. Gray, and S. K. S. Gupta. Author judgements about works they cite: three studies from psychology journals. *Social Studies of Science*, 25(3):477–498, 1995.

[49] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for Information Science*, 37:265–269, 1973.

[50] L. Smith. Citation analysis. *Library Trends*, 30(1), 1981.

[51] G. Taubes. Science journals go wired. *Nature*, 271:764–766, 1996.

[52] S. D. Whitehead. Auto-faq: An experiment in cyberspace leveraging. In C. Knoblock and A. Levy, editors, *Proceedings of the Second International WWW Conference*, volume 1, pages 25–38, Chicago, 1994.

[53] J. Yamron, I. Carp, L. Gillick, and S. Lowe P. van Mulbregt. A hidden Markov model approach to text segmentation and event tracking. In *Proceedings of the IEEE ICASSP*, 1998.

[54] B. Yuwono and D. L. Lee. Search and ranking algorithms for locating resources on the world wide web. In *ICDE*, pages 164–171, 1996.

[55] O. R. Zaiane and H. Jaiwei. Resource and knowledge discovery in global information systems: A preliminary design and experiment. In *Proceedings of Knowledge Database Discovery '95*, pages 331–336. 1995.